Abstract—Noncoherent decoding of trellis codes using multiple-symbol overlapped observations was shown previously to achieve close to the coherent performance. Optimal decoding by the Viterbi algorithm for L-symbol observations requires a number of states which grows exponentially with L. In this paper, two novel suboptimal algorithms are presented, for which the number of states is the same as the original code, yielding complexity depending weakly on L. For practical values of L, both algorithms are substantially less complex than the optimal algorithm. The first algorithm, the basic decision feedback algorithm (BDFA), is a low complexity feedback decoding scheme, based on the Viterbi algorithm. This algorithm is shown to suffer from increased error probability and from error propagation. A slight modification to this algorithm can, in most cases, reduce these effects significantly.

The second algorithm uses the BDFA as a basic building block. This algorithm is based on a novel concept called "estimated future" and its performance is very close to optimum for most practical cases with some additional complexity and memory requirements as compared to the first algorithm. Performance analysis and simulation results are also given.

I. INTRODUCTION

In a previous paper [1], the author introduced the notion of noncoherent trellis coded modulation (NTCM). The noncoherent decoding of multidimensional trellis coded modulation (TCM) on intersymbol interference (ISI) channels [3], [4] and for partial response (CPM) [5]. For some practical codes, degradation of 0.5-1 dB relative to the optimum is demonstrated.

The BDFA suffers from increased error event probability and from error propagation. However, by a small modification of the BDFA, we obtain another improved algorithm, which will be called modified decision feedback algorithm (MDFA). The second algorithm, the estimated-future decision feedback algorithm (EF DFA), which uses the BDFA as a basic building block, is based on a novel concept called "estimated-future," and performs very close to the optimal in most practical cases. Its degradation in high SNR (P_e < 10^{-3}) is negligible. Tight bounds on the performance of the EF DFA are given for two cases: 1) worst case error propagation and 2) the approximate model of geometrically distributed error propagation length.

The degradation of the suboptimal algorithms can be overcome by employing error detection and processing erroneous blocks off-line using an optimal algorithm. If the probability of error is low, the off-line process can be complex, since...
more time is available for its completion (causing delay in the decoding of that block).

The paper is organized as follows. In Section II, we give a review of the noncoherent decoding method. In Section III, we describe its implementation by the Viterbi algorithm. In Section IV, the BDFA is presented, followed by an explanation of its sources of degradation. An improvement of the BDFA, called MDFA, is outlined. The EFDFA is presented in Section V and followed by a performance analysis. In Section VI, we give simulation results which compare the various algorithms. A detailed step by step example of the EFDFA is given in the Appendix.

II. REVIEW OF THE IO-NMLSE

The solution for the problem of finding an optimal NMLSE depends on the statistics of the time variations of the carrier phase. When such statistics are unavailable, the derivation of the optimal NMLSE must start from some broad assumptions. We use the common assumption that the carrier phase is constant (but completely unknown) during some observation interval \((t, t + T)\) for any \(t\), but here, the observations are overlapping.

Based on the above, the IO-NMLSE discriminates between a set of possible transmitted waveforms \(\{x(i)(t)\}\) by choosing \(m\) which maximizes the following metric:

\[
\eta(x^{(m)}(t)) = \sum_{k=\infty}^{\infty} \left| \int_{t_k}^{t_{k+T}} r^*(t)x^{(m)}(t) \, dt \right|^2
\]

where \(r(t)\) is the received waveform (both \(x(t)\) and \(r(t)\) appear in the complex baseband representation), \(k\) is the observation number, \(\tau\) is the spacing between the beginnings of observations, and \(T\) is the observation length. In the digital implementation, where \(x^{(m)}(t)\) is a sequence of symbols of duration \(T_s\), the metric can be written as

\[
\eta(x^{(m)}) = \sum_{k=\infty}^{\infty} \left( \sum_{t=0}^{T_s-1} r^{(1)}_{k,t} x^{(m)}_{k,t} \right)^2 = \sum_{k=\infty}^{\infty} \Delta \eta_k
\]

where \(L\) is the observation length in symbols and \(\dagger\) denotes conjugate-transpose. Each branch in the trellis corresponds to an output waveform, which is represented by a complex vector, and is called “symbol.” For every symbol \(i, r_i\) is a complex vector given by the output of \(D\) complex matched filters, each for one complex dimension of modulation, \(x^{(m)}\) is the sequence of vectors of dimension \(D\), which is the signal space representation of \(x^{(m)}(t)\). Maximal amount of overlapping between consecutive observations is assumed.

The encoder uses trellis coded modulation which has the following parameters:

- \(N\) Number of states.
- \(B\) Number of input bits per branch.
- \(R = 2^B\) Number of branches reaching a node.

The trellis is assumed not to contain parallel transitions. However, only minor changes in the algorithms are required to remove this restriction.

III. OPTIMAL IO-NMLSE IMPLEMENTATION BY THE VITERBI ALGORITHM

It seems natural to choose the VA [7] for implementing the IO-NMLSE. However, the VA cannot be used without modification. We cannot use \(\Delta \eta_k\) as the branch metric since it is a function of the current branch value \(x_k\) together with the previous branch values \(\{x_{k-1}, \ldots, x_{k-L+1}\}\). Since the tentative decisions made by the VA should not affect the following branch metrics, this choice of metric will not cause optimal operation of the VA as a maximization algorithm. If we insist on using this metric in the VA, we get a suboptimal algorithm that will be described in the next section. In order to make the branch metrics independent of previous decisions, we can construct a new trellis diagram with \(NR^{L-1}\) states as follows.

A state in the new trellis will be assigned to each of the possible sequences of \(L\) consecutive states \(\{z_0, \ldots, z_{L-1}\}\) that can be produced by the original trellis. The original transitions from state \(z_{L-1}\) to state \(z_L\) are mapped to transitions from state \(\{z_0, \ldots, z_{L-1}\}\) to state \(\{z_1, \ldots, z_L\}\) for all possible choices of \(\{z_0, \ldots, z_L\}\). The corresponding branch value is the sequence of symbols \(\{x_0, \ldots, x_{L-1}\}\) which is the output of the path \(\{z_0, \ldots, z_{L-1}\}\) on the original trellis. For example, see Fig. 1. When using the new trellis, \(\Delta \eta_k\) is a function of the branch value only, enabling correct maximizations of the metric (2) by the VA.

Note that having \(NR^{L-1}\) states is sufficient but not necessary. For example, let us have a code with MPSK symbols having \(N/M\) states followed by a differential phase modulator (DPSK). The overall code has \(N\) states. Since only the phase differences are required for the noncoherent decoder, the VA can be written using only \(NR^{L-1}/M\) states. The same principle applies also to continuous phase modulated signals. There, the phase states can be eliminated [2].

If the complexity of the decoder is measured by the number of correlations per symbol, we get \(NR^2\) correlations. We would like to compare this with the number of correlations needed to implement the multiple symbol differential detector of Divsalar et al. [6]. There the number of states is not increased, but every \(L\) consecutive symbols (we ignore the overlapped phase shift keying (PSK) symbol for large \(L\))
We see that both complexities are asymptotically comparable. Also, since sliding correlation is performed, only one complex multiplication is needed per correlation instead of the augmented one. Doing so, the number states as well as the number of correlations stay constant as the memory, $K$, of the encoder. In our simulations we have used $M > 5K$, like in the VA, and the survivors have always converged as expected.

Since the BDFA is a recursive process, it contains provisions for the recursive update of $P^i_k$ and $J^i_k$. Denote the next state that follows the present state $s$ for the input symbol $0 \leq i < E$ by next$(s, i)$. Each path $P^i_k$ is appended with each one of the states $n_i = \text{next}(s, i), i = 0, \ldots, R - 1$, to form the candidate paths $P^i_{k+1} = \{P^i_{k+1}, P^i_{k+1-1}, \ldots, P^i_0\}$. There are $R$ paths $P^i_{k+1}$ which end in state $n$. For each of them, $J^i_{k+1}$ is computed by the formula

$$J^i_{k+1} = J^i_k + \sum_{j=0}^{L-1} r^i_{k+1-j}s^i_{k+1-j}$$

where $\{s^i_k\}$ are the output symbols of the path $P^i_k$. The index $i$ which maximizes $J^i_{k+1}$ is used to update $J^i_{k+1}$ and $P^i_{k+1}$ by $J^i_{k+1} = J^i_{k+1}^\star$ and $P^i_{k+1} = P^i_{k+1}^\star$ correspondingly.

This algorithm seems likely to result in a survivor path that has the largest metric. This is true most of the time, but not always. The VA assumes that when a decision is made, it does not depend on future decisions. This will not apply in our case. We are deciding $L - 1$ symbols too early. We should have decided on the branch of a state in time $t$, after we included the contribution of the symbols on the path emerging from this state continuing up to time $t + L - 1$. After time $t + L - 1$, there is no longer a dependence between the future symbols and the current decisions. To clarify, let $\tilde{P}_i$ be the transmitted path and let $\tilde{P}^i_{k+1}$ and $\tilde{P}^i_{k+1}$ be the candidate paths ending in state $n$ at time $t + 1$, where $n$ lies on the correct path (see Fig. 2). We extend these paths with the future states $\{\tilde{P}_{k+2}, \tilde{P}_{k+3}, \ldots\}$.

The optimal decision between $\tilde{P}^i_{k+1}$ and $\tilde{P}^i_{k+1}$ is

$$\max \{\eta(\tilde{P}^i_{k+1})_{\infty}, \eta(\tilde{P}^i_{k+1})_{\infty}\} = \max \{\eta(\tilde{P}^i_{k+1})_{\infty}, \eta(\tilde{P}^i_{k+1})_{\infty}\}.$$

In practice, the required sequence of states $\{\tilde{P}_{k+2}, \tilde{P}_{k+3}, \ldots, \tilde{P}_{k+L}\}$ is unknown to the decoder and will be called the future path. The next two algorithms, the MDFA and the EF DFA, are based on estimations related to this future path.
If we want to make the correct decision, we may want to include the future path in the comparison. But, for every state, there are $R^{L-1}$ possible future paths which we will have to consider (or at least consider the most probable one, as will be done in the EF DFA). Thus, the complexity of the decoder is increased at least by $R^{L-1}$, the same as in the VA. Alternatively, we can convert the encoder to an equivalent one by adding $L-1$ unconnected stages to the encoder shift register. When such encoder is used, in every decision the candidate paths contain the same last $L-1$ symbols. Given any future path, its contribution to all the candidates is equal so it cannot effect the decision. In this case the BDFA makes optimal decisions. However, the this encoder has $R^{L-1}N$ states and, moreover, it can be shown that it is essentially equivalent to the augmented trellis of Section III.

The suboptimal algorithm causes degradation through two processes. The first is an increase in the error event probability and the second is error propagation. The increase in the error event probability can be explained as follows. Each time we make a decision we essentially compare two truncated sequences instead of infinite sequences, as in the IO-NMLSE. The sequences are truncated at the current symbol when we compare and make the decision and the future path’s contribution is ignored. The first error event probability of the BDFA can be evaluated by the union bound (see [1]) when using the truncated sequences instead of infinite sequences.

Let us now discuss the error propagation (to be abbreviated as EP), which is the main cause of degradation. After an error occurs in the decision process, the error probability for future decisions increases. If a new error occurs, further future decisions may be in error in a chain reaction-like fashion. This process of EP leads to large error bursts. For some applications this phenomena is not crucial. For example if a message must be transmitted without error, and any number of errors cause the message to be discarded and retransmitted.

The mechanism of the EP is explained by referring to Fig. 3. Suppose that path 1 was transmitted. At point $B$ a wrong decision has been made (due to the noise), i.e., $y_1 < y_2$. Suppose that $y_1 < y_2$. Returning to the algorithm, at point $A$ we now decide between path 3 and path 2. Path 3 might win even though it would have lost had it been compared to path 1 at point A. Even though path 1 has the maximal metric up to point A, path 3 has won due to the wrong decision made at point $B$. Path 3 can deviate from the true path by more symbols than the original error, i.e., path 2, and still win. After making the first wrong decision, candidate paths will not have to win over the maximal path but over a path whose metric is less than the maximal. The EP can continue as long as there are candidates that pass the lower threshold. It is very difficult to predict when this process stops.

Let us investigate the case where we can use the future symbols of the path in the decisions. In this case we argue that even if we make an error, no EP will result. Let us suppose that we could have decided at point $B$ which of the sequences 1 or 2 were transmitted using $\eta(\cdot)_B$ instead of $\eta(\cdot)_B$. Suppose that path 2 won again. In this case no EP can occur because

![Fig. 3. Explanation of the error propagation. Path 3 may win at point A due to an error that was made at point B.](image)

when path 2 was decided at point $B$ (although it is an error), path 2 had indeed the largest metric (up to point $A$) so we are truly maximizing the path metric between these three paths. Note that if point $A$ is less than $L-1$ symbols away from point $B$, EP can still occur in later decisions. An important and not very intuitive fact is that for the same input, error events that occur both in the optimal algorithm and in the BDFA will not cause EP in the BDFA.

In the analysis of the following algorithm (EF DFA), we will assume that the error propagation length can be modeled by a geometric distribution. This model results from the assumption that whenever we are already in an EP event, the probability that it will stop is the same no matter when this EP began. In other words the EP is assumed to be memoryless. This model holds well for large EP events. The validity of the model is demonstrated in Fig. 4, where the model fits the measured data taken by simulation of the BDFA.

**Modified Decision Feedback Algorithm (MDFA)**

We can improve the decision process of the BDFA even without knowing the future $L-1$ states emerging from the current state. Let $\mathbf{x}_i$ be the transmitted symbols. Let us assume that the symbols $\mathbf{x}_{t+2}, \cdots, \mathbf{x}_{t+L}$ are known to the decoder when it has to make a decision at time $t+1$, see Fig. 2. If state $n$ is on the correct path, the optimal decision rule is then the following. For each of the new paths $P_{t+1}^{k,n}$ we compute $\hat{j}_{t+1}^{k,n}$ by

$$\hat{j}_{t+1}^{k,n} = J_t^{k,n} + \sum_{k=t+1}^{t+L} \left| \sum_{j=0}^{L-1} \beta_{k-j} \hat{s}_{t+j}^{k,n} \right|^2$$  \hfill (8)
where $\hat{x}_j^{s_i:n} = \hat{x}_j$ (the known symbols) if $j > t + 1$. If the next state $n$ where we make the decision is not on the correct path, then there is no meaning to this expression, and using it may cause wrong decisions (w.r.t. the IO-NMLSE). Such a wrong decision can only change the paths competing with the correct path in later decisions. For example, in Fig. 2, wrong decision at point $Q_1$ will change the candidate path competing with the correct path at point $Q_2$. Assuming that the correct path has the largest metric, no other path will win above it no matter which path it is. Thus the decoding error probability will not increase.

We can express the received signal as $r_j = \alpha \hat{x}_j e^{j\theta} + n_j$ where $\theta$ is the carrier phase and $\alpha$ is the channel attenuation (both assumed constant over $2L$ symbols, but unknown). Since we use constant amplitude symbols such that $\hat{x}_j = 1$

$$r_j^* \hat{x}_j = \alpha e^{-j\theta} + n_j \hat{x}_j, \quad j = t - L + 2, \cdots, t + L. \quad (9)$$

For the correct path ($n$ is correct and $s_i$ is correct, and no errors, $\hat{x}_j^{s_i:n}$ for $j = t - L + 1, \cdots, t + 1$ are the correct symbols. Let us define

$$\mu = \sum_{j=t-L+1}^{t+1} r_j^* \hat{x}_j^{s_i:n} = L \alpha e^{-j\theta} + \sum_{j=t-L+1}^{t+1} n_j \hat{x}_j^{s_i:n}. \quad (10)$$

Provided that the SNR in $L$ symbols is high enough, $\alpha e^{-j\theta}$ can be estimated by $\hat{\alpha}$. Then we can use $\hat{\alpha}$ as an estimate for $r_j^* \hat{x}_j$, whenever $j > t + 1$ and use it in (8). We tested several other ways to estimate $\alpha e^{-j\theta}$ but none were more successful than this simple method. To summarize, the change in the algorithm is the following: Compute

$$\mu = \sum_{j=0}^{L-1} r_{t+1-j}^* \hat{x}_j^{s_i:n} \quad (11)$$

$$\hat{J}_{t+1}^{s_i:n} = J_t^{s_i:n} + |\mu|^2 \quad (12)$$

and

$$\hat{J}_{t+L}^{s_i:n} = J_{t+1}^{s_i:n} + \sum_{k=t+1}^{t+L} \left\{ \sum_{j=0}^{L-1} \left\{ \frac{r_j^* \hat{x}_j^{s_i:n}}{L}, \text{if } k-j \leq t+1 \right\} \right\}^2 \quad (13)$$

then $\hat{J}_{t+L}^{s_i:n}$ is used as an estimate for $J_{t+L}^{s_i:n}$. The index $i$ which maximizes $\hat{J}_{t+1}^{s_i:n}$ is used to update $J_{t+1}^{s_i:n}$ and $P_{t+1}^{s_i:n}$ by $J_{t+1}^{s_i:n} = \hat{J}_{t+1}^{s_i:n}$ and $P_{t+1}^{s_i:n} = P_{t+1}^{s_i:n}$. The algorithm was found to reduce the number of error propagation events significantly. Once an EP has begun, the efficiency of the algorithm decreases since then the estimate $\hat{\alpha}$ is not as good. Nevertheless, it can still reduce significantly the average length of the EP events, as was confirmed in our simulations.

V. ESTIMATED-FUTURE DECISION FEEDBACK ALGORITHM

The MDFA still has degradation compared to the optimal algorithm. Thus a better algorithm is desired. A novel algorithm, which performs very close to the optimal algorithm, but with significantly lower complexity, was found. This algorithm, like the BDFA, uses the original code trellis. On the other hand, the optimal algorithm uses an augmented trellis with a large number of states. The EFDFA complexity is roughly four times that of the BDFA.

The algorithm uses a novel concept called estimated-future to improve the decision process. We have previously recognized that we need to include the future path to make the current decision in order to make optimal decisions in the BDFA. If such a future path is given, but it is not completely reliable, we call it estimated-future. The algorithm works as follows. In each trellis state, at each time, two independent decisions are being made. One is called the no-future (NF) decision, which is similar to the BDFA, and the other is a decision using the $L - 1$ estimated-future symbols and it is called the with-future (WF) decision. The first suffers from FP and increased sequence error as discussed in the previous section. The second will make the best decisions as long as the future sequence which is being used is the true one. On the other hand, if wrong future is used for the WF decision, then it can cause an error. However, utilizing both the NF and the WF decisions, one can arrive at a combined decision, which fails only when both NF and WF fails.

How can one possibly know the future? The approach is to save a block of the input signal in memory and perform a DFA (BDFA or MDFA) backwards, starting from the end of the block. After the backward process ends, we have the survivor paths belonging to each state at each time in the trellis within the block. These paths will be used as estimate future paths. The future estimation performance by this method is as good as the performance of the reversed code (reversing the generators) when decoded with a DFA. The performances of the code and its reversed version can be different only because of the suboptimality of the DFA.

The input stream is divided into overlapping blocks, each block having a length of $A + W$ symbols and starts $A$ symbols after the beginning of the previous block (see Fig. 5). The backward process (BP) operates first and processes the whole block. Then the forward process (FP) operates on the first $A$ symbols of the block. The FP is continuous from block to block. The blocking is intended only for the operation of the BP. The section of length $W$ is intended for letting the BP converge from the initial conditions, in which no particular state is used as a beginning state. This convergence region can be eliminated by inserting $K + B(L - 2)$ known input bits at the end of each block of $BA$ input bits, where $K$ is the code memory, and in that way reach a known state at the end of each block.

The FP and the BP operating on the same block are highly correlated. When there is an error event in the FP, a similar
error event is also likely to occur in the BP. However, the initial error and the EP following it in the BP is going in the other direction—towards the past, leaving the estimated-future intact at the time needed for correcting the forward EP (see Fig. 6).

Returning to the issue of convergence of the BP, the first decisions in the BP are made using very few symbols. In particular, the first decision is based only on the last symbol in the block (remember, we are going backwards so this last symbol is our first). This means that it is probable that we start with an error. Since the BP suffers from EP, the convergence time is similar to the time needed to recover from error EP. This is why we recommend the use of MDFA for the BP instead of BDFA.

For the algorithm description that follows, we are using the BDFA decisions in both the backward decisions and the forward NF decisions. Each one of them can be replaced by the modified version without altering the principle of operation.

A. The Backward Process

The BP operates on a block of length \( A + W, kA \leq t < (k + 1)A + W \). This is exactly the BDFA going back in time. For each state \( s \), the algorithm keeps track of the associated survivor, i.e., the most likely path of previous states

\[
Q_t^s = \{q_{l,t-k}, q_{l,t-k-1}, \ldots, q_{l,t-1}, q_{t, t}\}
\]

and also of an accumulated metric

\[
E_t^s = \sum_{k=L}^{(k+1)A+W} \left[ \sum_{j=1}^L r_{k+j} x_{k+j}^s \right]^2
\]

where \( x_{k+j}^s \) is the output of the trellis branch connecting \( q_{l,t-k} \) with \( q_{l,t-1}^s \). Denote the previous state which produced \( s \) by moving one step forward in the code trellis for the input symbol \( 0 \leq i < R \) by previous \( (s,i) \). For the recursive update of \( Q_t^s \) and \( E_t^s \), each path \( Q_t^s \) is appended with each one of the states \( p_i = \text{previous}(s,i)\) to form the candidate paths

\[
Q_{t+1}^{s+1} = \{q_{l,t-k+1} + W, q_{l,k+1}^s + W, \ldots, q_{l,t-1}^s + W, q_{t, t+1}^s \}
\]

There are \( R \) paths \( Q_{t+1}^{s+1} \) which end at state \( p_i \). For each of them, \( E_{t+1}^{s+1} \) is computed by the formula

\[
E_{t+1}^{s+1} = E_t^s + \sum_{k=0}^{t+L-1} r_{k+j}^s x_{k+j}^{s+1}
\]

where \( \{x_{k+j}^{s+1} \} \) are the output symbols of the path \( Q_{t+1}^{s+1} \). The index \( i \) which maximizes \( E_{t+1}^{s+1} \) is used to update \( E_{t+1}^{s+1} \) and \( Q_{t+1}^{s+1} \).

B. The Forward Process

Given the estimated-future (contained in \( Q \)), the FP works as follows.

For each state \( s \), the algorithm keeps track of two associated survivors. The first, \( C_t^s \), is called the NF survivor and is our best guess of the minimum likelihood (ML) path

\[
C_t^s = \{q_{l,t-k}, q_{l,t-k+1}, \ldots, q_{l,t-1}, q_{t, t}\}, \quad s \neq s
\]

Here \( c_{l,t} \) denotes the state indexed by \( k \) in the list \( C_t^s \).

For every path \( C_t^s \) there is an associated accumulator metric

\[
G_t^s = \eta(C_t^s)
\]

The second survivor, \( F_t^s \), is called the WF survivor and is used as a temporary variable

\[
F_t^s = \{q_{l,t-k}, q_{l,t-k+1}, \ldots, q_{l,t-1}, q_{t, t}\}
\]

Its associated accumulator metric is

\[
H_t^s = \eta(F_t^s)
\]

The second survivor, \( F_t^s \), is the path that includes the estimated-future. It extends from the state \( s \) at time \( t \) to \( L-1 \) symbols, towards the future. As in the implementation of the VA, only \( M \) (truncation length) last states need to be saved in each path list. The algorithm works recursively. Each time we do the following.

Step 1: For each state \( s \), form the WF candidate path \( F_t^{s,m} = q_{t+1}^s \) by appending the state \( q_{t+1}^s \) to \( F_t^s \).

Step 2: For each \( F_t^{s,m} \), compute the accumulated metric \( H_t^{s,m} \) by

\[
H_t^{s,m} = \eta(F_t^{s,m}) = H_t^s + \sum_{j=0}^{t+L-1} r_{j}^s y_{j}^{s,m}
\]

where \( \{y_{j}^{s,m} \} \) denotes the output symbols of \( F_t^{s,m} \).

Step 3: For each state \( s \), find the next states \( n_i = \text{next}(s,i) \) for \( 0 = 0, \ldots, R-1 \). For every \( n_i \neq m \) form the WF candidate path \( F_t^{n_i} \) by appending the sequence \( \{q_{l,t+1+1}, q_{l,t+1+2}, q_{l+1+3}, \ldots, q_{l,t+1+L} \} \) to \( C_t^s \).

Step 4: For each \( F_t^{n_i} \), compute the accumulated metric \( H_t^{n_i} \) by

\[
H_t^{n_i} = \eta(F_t^{n_i}) = G_t^s + \sum_{k=0}^{t+L-1} r_{k-j}^{n_i} y_{k-j}^{n_i}
\]

where \( \{y_{k-j}^{n_i} \} \) denotes the output symbols of \( F_t^{n_i} \).

Step 5 (WF Decision): For each state \( w \), there is a total of \( R \) values \( H_t^{n_i} \). The index \( i \) which maximizes \( H_t^{n_i} \) is used to update \( H_t^s \) and \( F_t^s \) by \( H_t^{n_i} = H_t^{n_i} \) and \( F_t^{n_i} = F_t^{n_i} \).

Step 6: For each state \( s \) and input \( i = 0, \ldots, R-1 \), form the NF candidate path \( C_{t+1}^{n_i} \), by appending the state \( n_i = \text{next}(s,i) \) to \( C_t^s \).
Step 7: For each $C_{t+1}^{*,n_1}$, compute the accumulated metric $G_{t+1}^{*,n_1}$ by

$$G_{t+1}^{*,n_1} = G_t + \left| \sum_{j=0}^{L-1} r_{t+1-j} C_{t+1-j}^{*,n_1} \right|^2$$

where $\{ C_{t+1-j}^{*,n_1} \}$ denotes the output symbols of the path $C_{t+1-j}^{*,n_1}$.

Step 8 (NF Decision): For each state $w$, find all states $u_j, j = 0, \ldots, l - 1$, such that the path $F_{t-1}^{u_j}$ ends with $w$, i.e., $q_{e-1+2,t+1} = w$. For an empty set, $i = 0$. Including the $R$ values of the form $C_{t+1}^{*,w}$, we define

$$\alpha_i = \begin{cases} G_{t+1,w} & \text{if } i < R \\ H_{t+1-L+2} & \text{if } R \leq i < R + l \\ 0 & \text{if } i = 0, \ldots, R + l - 1. \end{cases}$$

The index $i$ which maximizes $\alpha_i$ is used to update $C_{t+1}^{*}$ and $G_{t+1}$ by $G_{t+1}^{*,w} = \alpha_i$ and

$$C_{t+1}^{*,w} = \begin{cases} C_{t+1}^{*,w} & \text{if } i < R \\ F_{t+1}^{*(L+2)} & \text{if } R \leq i < R + l \end{cases}$$

Step 9: Find the $w$ which maximizes $G_{t+1}^{*,w}$. The input data which corresponds to the branch connecting the two "old" states $C_{t+1,i+M+1}$ and $C_{t+1,i+M+1}$ serves as the decoder output, where $M$ is the decoder memory length.

The operation of the algorithm is as follows (refer also to Fig. 7). $G_{t}$ always holds the best survivor path that we have at time $t$ to state $s$. Its update is done in Step 8, the NF decision, where two kinds of competitors are compared: The NF candidate paths which are extensions by one symbol of the previous NF survivor and are not using the estimated-future, and the WF candidate paths which are found in $F_t$. The WF candidate paths are the survivors of past decisions that used the estimated-future paths. These decisions were made at time $t - L + 1$, while the path is used at time $t$. When the WF survivor is used, the $L - 1$ states that used to be the future at time $t - L + 1$ are the past for time $t$, and the last state becomes the current one. In case the future estimation was correct, that WF path is a result of an optimal decision. In this case this path will either win or its metric will be equal to the maximal NF path metric. A win condition indicates that wrong decisions have been made which might lead to EP, unless corrected by the winning path from $F_t$. Equality between the WF path and the NF candidate indicates that the previous NF decisions were correct leading to the same survivor. Correct decisions mean that the decisions are leading to the ML path, not necessarily to the transmitted path.

In order to update $F_t$, we make the decisions about the survivor path at time $t$, using the future $L - 1$ symbols which are assumed to be correct. The candidates that pass through the states $s$ at time $t$ are divided into two categories: The first category contains paths that are extensions of previous WF paths, by appending the next state from the estimated-future path (Step 1), like path 1 in Fig. 7. The second category contains NF paths combined with the estimated-future path (Step 3), like path 2 and path 3 in Fig. 7. The paths of the second category which share their last $L$ states with paths of the first category, like path 3, are eliminated from the comparison since those two candidates have already been compared in the previous WF decision. This elimination is taking place in Step 3 checking that the next state $s_{t+1}$ is not equal to the next state from the estimated-future, $s_{t+1}$.

The effect of the wrong future paths is to add candidates to the comparison at Step 8, the NF decision. These candidates cannot be defined, win over the correct (ML) candidate. Thus, no error will result.

C. Performance Analysis

First, let us introduce several variables. Let $p_e$ and $N_e$ denote the probability per symbol and the average number of bits in error, respectively, of an error event of the 10-NMSE. Let $p_f$ denote the first event error probability of the forward DFA (or the NF decisions) and $p_b$ the first event error probability of the backward DFA. $p'_f$ is the probability of a correctable forward error event, i.e., this error can be corrected by using a correct estimated-future. Errors are uncorrectable only if they are also made by the optimal decisions. From the union bound, $p'_f \geq p_f - p_e$. Let $\gamma$ denote the conditional probability that EP occurs in the BP given the first error occurred. $\gamma_f$ denotes the EP conditional probability given a correctable forward error event occurred. Let $N_f$ be the average number of bit errors in a correctable forward first error event (first meaning that it is not a part of EP).

We will use the following assumptions. For the worst case, an incorrect future path will prohibit correction of NF errors (any NF error which is the result of not making the optimal decision is corrected by the algorithm when a correct estimated-future is used, but also can sometimes be corrected by using an incorrect estimated-future). EP continues until a correct future is encountered, then stops within a
few symbols. For the derivation it is assumed that the EP stops immediately when the correct future is encountered. We assume that \( p_b, p_f, p_e < 1/4 \), so there is at most one error per block. \( A \gg L, K \) so that edge effects can be ignored. Finally, we assume that if a convergence region \( W \) is used for the BP, it converges before the end of the block \( A \).

1) Worst-Case Model for Error Propagation: We begin with the simplest, but worst-case, model for the EP. It assumes endless EP until the end of the block. The resulting bound for the error event probability per symbol, \( P_{\text{seq}} \), is

\[
P_{\text{seq}} = \frac{P_A}{A} \leq p_e + (L - 1)(1 - \gamma_b)p_fp_b + \frac{1}{A} \sum_{i=0}^{A-1} p_f p_b \gamma_b (A - i) \frac{A^2}{2} + \frac{A}{2} p_f p_b \gamma_b
\]

(25)

where \( P_A \) is the block error probability. The second term is the contribution of the uncorrected forward errors due to backward errors that occur in the near future (< \( L \) symbols ahead) and does not cause EP. The third term reflects the uncorrected forward errors which are not corrected due to EP in the BP, one which originated in the region \( i < t < A \) of the block. For the bit error probability we get the following expression:

\[
P_{\text{bit}} \leq \frac{p_e N_e}{B} + \frac{1}{A B} \sum_{i=0}^{A-1} p_b \gamma_b (A - i) p_f + \frac{N_f'}{B} \left( \frac{A^2}{2} + \frac{A^2}{12} \right) + p_b \gamma_b p_f \gamma_b p_b \frac{N_f' A}{B}.
\]

(26)

\((A - i)/2\) is the average number of symbols in the forward EP. This EP starts at time \( i \) and ends at the first backward error. Its location is uniformly distributed leading to the reduction of the EP length by half. For close to optimal operation, the second term should be as small as possible. On the other hand, if framed transmission is used, then we have to keep \( A \gg L + K \) to minimize degradation due to throughput reduction. If a convergence region is used, then \( A \gg W \) is needed to validate the convergence assumption. Otherwise, backward errors originating in that region will dominate. Thus an optimum value for \( A \) exists.

2) The Refined Model: By using a probabilistic model for the EP length, we can get results which are closer to the actual performance. As mentioned previously, we will model the EP length by a geometric distribution with mean \( 1/\lambda \) (specifically \( \lambda_f \) for forwards and \( \lambda_b \) for backwards). With this model, the probability that the EP length (excluding the first error) is equal to \( k \) is \( \gamma (1 - \lambda)^{k-1} \). The probability that at time \( i \) we are in a backward error is

\[
\Pr(\text{iis in error}) = p_b (1 - \gamma_b) + p_b \gamma_b \sum_{j=i+1}^{A-1} (1 - \lambda_b)^{j-i-1}
\]

(27)

\[
= p_b (1 - \gamma_b) + \frac{p_b \gamma_b}{\lambda_b} [1 - (1 - \lambda_b)^{A-i-1}].
\]

We can investigate two possible limiting cases. The first is the case of \( A \gg 1/\lambda_f, 1/\lambda_b \) which means that the block length is much larger than the EP average length. The second limit will be the opposite case where the block length is much shorter.
than the average EP length. This is equivalent to the worst-case EP model. For the first case $A \gg 1/\lambda_b$, $1/\lambda_f$.

Taking the limit $A \to \infty$ in (30) gives

$$P_{\text{seq}} \leq p_e + (L-1)p_f p_b + \frac{p_f p_b \gamma_b}{\lambda_b}. \quad (34)$$

The result can be explained as follows. The probability that a particular decision will be a part of a backward EP is $p_b \gamma_b / \lambda_b$. This is the result one obtains for any physical problem of memoryless arrivals of events with memoryless distributed time duration (for example, the model used for the availability of telephone lines). Taking $A \to 0$ in (32) we get

$$P_{\text{bit}} \leq \frac{p_e N_e}{B} + (L-1)p_fp_b \left[ \frac{N_f}{B} + \frac{L}{4} \gamma_f (1 - \gamma_b) \right]$$

$$+ \frac{1}{\lambda_b} p_f \gamma_f \gamma_b \frac{N_f}{B} + \frac{p_f \gamma_f \gamma_b \lambda + \lambda_f}{\lambda_b (\lambda_b + \lambda_f) - (\lambda_b - \lambda_f)^2}$$

$$\mathcal{E} \approx \frac{p_e N_e}{B} + \frac{p_f \gamma_f \gamma_b \lambda + \lambda_f}{\lambda_b (\lambda_b + \lambda_f) - (\lambda_b - \lambda_f)^2}. \quad (35)$$

Here the probability of forward EP, $p_f \gamma_f \gamma_b \lambda_b / \lambda_b$, is multiplied by the average length of the EP which is $1/\lambda_b + \lambda_f$, the average minimum lifetime of two processes of rate $\lambda_b$ and $\lambda_f$.

The second limiting case is $A \ll 1/\lambda_b, 1/\lambda_f$. In this case the result should be the same as the worst case EP case. We take the limit as $\lambda_f, \lambda_b \to 0$ and get

$$P_{\text{seq}} \leq p_e + \frac{A - 1}{2} p_f \gamma_f \gamma_b \lambda + \frac{A + 1}{2} p_f \gamma_f \gamma_b \lambda. \quad (36)$$

and

$$P_{\text{bit}} \leq \frac{p_e N_e}{B} + (L-1)p_fp_b \left[ \frac{N_f}{B} + \frac{L}{4} \gamma_f (1 - \gamma_b) \right]$$

$$+ \frac{p_f \gamma_f \gamma_b \lambda + \lambda_f}{\lambda_b (\lambda_b + \lambda_f) - (\lambda_b - \lambda_f)^2}$$

$$\mathcal{E} \approx \frac{p_e N_e}{B} + \frac{p_f \gamma_f \gamma_b \lambda + \lambda_f}{\lambda_b (\lambda_b + \lambda_f) - (\lambda_b - \lambda_f)^2}. \quad (37)$$

$$P_{\text{bit}} < \frac{p_e N_e}{B} + (L-1)p_fp_b \left[ \frac{N_f}{B} + \frac{L}{4} \gamma_f (1 - \gamma_b) \right] + \frac{p_f \gamma_f \gamma_b \lambda + \lambda_f}{\lambda_b (\lambda_b + \lambda_f) - (\lambda_b - \lambda_f)^2}$$

$$\mathcal{E} \approx \frac{p_e N_e}{B} + \frac{p_f \gamma_f \gamma_b \lambda + \lambda_f}{\lambda_b (\lambda_b + \lambda_f) - (\lambda_b - \lambda_f)^2}. \quad (32)$$

---

**Example 1:** Let us investigate the performance of the algorithm for the code of rate $1/2$, four states code with BPSK modulation with $L = 3$ and $E_b/N_0 = 5$ dB. By the BDFA simulation program, we have measured the following parameter values (the code is symmetric so all backward and forward parameters are the same):

$$p_e = 0.00017, \quad p_b = p_f = 0.002, \quad \frac{1}{\lambda_f} = \frac{1}{\lambda_b} = 19, \quad \gamma_f = \gamma_b = 0.845, \quad N_e = 1.72.$$  

Since $p_f \gg p_e$, the majority of the forward errors are correctable and we can assume $p_f' = p_f, \gamma_f' = \gamma_f$ and $N_f' = N_f$. In addition, we can assume $N_f = N_e$. By simulating the above parameters and choosing $A = 50$ and the framed input option, we get

$$P_{\text{seq}} = 0.00022, \quad P_{\text{bit}} = 0.00056.$$  

The actual values measured by simulation are follows:

$$P_{\text{seq}} = 0.00021, \quad P_{\text{bit}} = 0.00049.$$  

---

**VI. SIMULATION RESULTS**

The algorithms presented were evaluated by simulation (see Fig. 8). The channel used was AWGN with a slowly varying phase (up to $10^\circ$ change over one observation of length $L$ symbols). The codes used were the quaternary ($R = 4$) linear noncoherent coded modulation (LNCM) family [1]. These codes are built over the modulo four group and use QPSK symbols. Both have rate $1/2$. The first code has 16 states and generators (in base four) 133, 231. The second has 64 states and generators 2123, 1312. The codes are maximum $d_{\text{free}}$ codes with $d_{\text{free}}$ of seven and 10, respectively. For proper operation of the MDFA, it is necessary to have enough SNR per observation. For $E_b/N_0 = 2$ dB, we need $L \geq 5$ and for $E_b/N_0 = 3$ dB, we need $L \geq 4$. $L = 4$ was used in the 16 states code simulation and $L = 5$ for the 64 states code. We chose $W = 100$ for the EF DFA to achieve a good convergence. It was convenient to choose also $A = 100$.

For the 16 states code, 1024 states were required for the optimal algorithm, where only 16 states are required in the suboptimal algorithms. For the 64 states example, 16 384 states are required, so the optimal algorithm could not be simulated. Since the suboptimal algorithms require only 64 states, their complexity is two orders of magnitude lower in this example. Compared to the BDFA, the CPU time requirements of the EF DFA is four to five times greater.
The union bound was derived in [1] and is a tight upper bound on the performance of the optimal implementation of the IO-NMLSE. In the bit error performance curves, due mainly to EP effects, the BDFA lost 2 dB compared to the optimal. The MDFA worked well, but degradation of up to 0.5 dB in one case and 0.8 dB in the other remains. The EFDFA algorithm essentially achieved the optimal performance in moderate and high SNR ($P_b < 10^{-3}$). In low SNR ($P_b \approx 10^{-2}$), we see 0.2–0.3 dB degradation. The reason for the degradation in low SNR is that the ratio $P_b P_f / P_e$ becomes higher.

Examining the error event performance curves, we see that all the algorithms suffer less degradation compared to the bit error rate case. The reason is that all the algorithms suffer from increased length error bursts when they fail. We also note that the modification in the MDFA reduces the error events probability.

VII. CONCLUSION

Several algorithms for implementing the decoder for NCM were introduced. Their performance was explained and evaluated analytically and via computer simulations. The BDFA, although simple to implement, is not recommended for actual use, since with only a slight increase in complexity we can implement the MDFA which can reduce the degradation to less than 0.5 dB (depending on the code). The EFDFA algorithm essentially achieved the optimal performance in moderate and high SNR ($P_b < 10^{-3}$). Its complexity for a practical case is two order of magnitude lower than that of the optimal implementation. Since the IO-NMLSE performance is very close to that of the coherent MLSE, the algorithm shows a practical noncoherent decoding which performs very close to the optimal coherent one.

APPENDIX

DETAILED EXAMPLE OF THE EFDFA OPERATION

We would like to decode a rate $\frac{3}{2}$, $K = 3$ code with BPSK modulation, and we choose $L = 3$ (see Fig. 9). The input to the decoder is as shown, where the numbers are real for simplicity. The symbols are in general complex vectors.
The initial conditions are as follows: 0 is received for \( t < 0 \) and the state of the encoder is 0 at time \( t = -1 \), where all paths originate. For \( t < 2 \) we only compute the metrics—there are no decisions.

At \( t = 0 \): \( \delta^0 = \{0,0,0,0\}, \bar{H}_0^1 = 10.74, F_0^1 = \{0,1,2,0\}, \bar{H}_0^1 = 26.1, C_0^0 = \{0,0\}, C_0^0 = \{0,1\}, G_0^0 = 4 \).

Steps 1 and 2: \( s = 0 \to m = 0 \to s = 1 \to m = 2 \). \( F_1^0 = \{0,0,0,0\}, \bar{H}_1^0 = 10.74 + (0.5 + 0.8 - 0.9 - 0.9 - 1 - 1)^2 = 16.99, F_1^1 = \{0,1,2,0\}, \bar{H}_1^1 = 26.1 + (-0.5 + 0.8 + 0.9 + 0.9 - 1 - 1)^2 = 26.11.

Steps 3 and 4: For \( s = 0 \) and \( \{n_t\} = \{0,1\}, m = 0 \) so we take only \( n_t = 1 \). \( F_1^0 = \{0,0,0,0\}, F_1^1 = \{0,0,1,2\} \).

Steps 5 and 7: \( C_1^1 = \{0,0,0\}, C_1^0 = G_0^0 + (-0.5 - 1.5 + 0.5 + 0.8)^2 = 4.49, C_2^1 = \{0,0,0\}, C_2^0 = G_0^0 + (-0.5 - 1.5 - 0.5 - 0.8)^2 = 14.89, C_1^1 = \{0,0,1\} \).

Step 8: \( C_2^1 = \{0,0,0\}, C_2^0 = \{0,0,1\} \).

The path that involves \( F_{t - 2} + F_{t - 2} \) is not relevant since \( F \) is not defined at \( t = -1 \).

At \( t = 1 \): Steps 1 and 2: \( s = 0 \to m = 0 \to s = 1 \to m = 2 \). \( F_2^0 = \{0,0,0,0,0\}, \bar{H}_2^0 = 50.63, F_2^2 = \{0,0,1,2,0\}, \bar{H}_2^2 = 26.27, F_2^1 = \{0,1,2,0,0\}, \bar{H}_2^1 = 12.67.

Steps 3 and 4: \( C_2^0 = \{0,0,0,0\}, C_2^0 = \{0,0,1,2\} \).

Steps 5: \( \bar{H}_2^0 > \bar{H}_2^1 > \bar{H}_2^0 > \bar{H}_2^0 > \bar{H}_2^0 > \bar{H}_2^0 > \bar{H}_2^0 > \bar{H}_2^0 > \bar{H}_2^0 > \bar{H}_2^2 \).

Step 7: \( F_2^0 = \{0,0,0,0,0,0\}, H_2^0 = 50.63, F_2^2 = \{0,0,1,2,0,0\}, H_2^2 = 26.27, F_2^1 = \{0,1,2,0,0,0\}, H_2^1 = 12.67.

At \( t = 3 \): We omit Steps 1–5.

Steps 6 and 7: We are interested only in the decision at state 0: \( C_4^0 = \{0,0,1,2,0,0\}, C_4^0 = 26.27 \).
\(C^2_4 = \{0, 0, 1, 3, 2, 0\}, G^2_4 = 31.47, \alpha_4 = \{G^0_4, G^2_4, H^0_2, H^1_2, H^2_2, H^3_2\} = \{26.27, 31.47, 50.63, 29.75, 26.27, 31.47\}\). The maximal is \(H^3_2\), so we have \(C^0_4 = F^0_2 = \{0, 0, 0, 0, 0, 0\}, G^0_4 = H^3_2 = 50.63\). Eventually the correct path is decided on and overrides the wrong decisions of the past. Note that without this help from the WF candidate, the NF decision would have made another wrong decision. This is a part of an endless EP that occurs in this example if the estimated-future is not used or is wrong and the input remains \(-1\).

REFERENCES


Dan Raphaeli (M'92), for a photograph and biography, see p. 183 of the February issue of this TRANSACTIONS.